PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 15/18 — A1

(11) International Publication Number: WO 97/44741

(43) International Publication Date: 27 November 1997 (27.11.97)

(21) International Application Number: PCT/US97/08951

(22) International Filing Date: 23 May 1997 (23.05.97)

(30) Priority Data:
60/018,191 23 May 1996 (23.05.96) US

(71) Applicant *(for all designated States except US)*: THE UNITED STATES OF AMERICA, represented by THE SECRETARY, DEPARTMENT OF HEALTH AND HUMAN SERVICES, THE NATIONAL INSTITUTES OF HEALTH [US/US]; Office of Technology Transfer, Suite 325, 6011 Executive Boulevard, Rockville, MD 20852-3804 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: HUNTER, Lawrence, E. [US/US]; 12820-A River Road, Potomac, MD 20854 (US).

(74) Agent: NOONAN, William, D.; Klarquist, Sparkman, Campbell, Leigh & Whinston, LLP, One World Trade Center, Suite 1600, 121 S.W. Salmon Street, Portland, OR 97204 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published

With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SYSTEM AND METHOD FOR COMBINING MULTIPLE LEARNING AGENTS TO PRODUCE A PREDICTION METHOD

(57) Abstract

System and method for improving the performance of learning agents such as neural networks, genetic algorithms and decision trees that derive prediction methods from a training set of data. In part of the method, a population of learning agents of different classes is trained on the data set, each agent producing in response a prediction method based on the agent's input representation. Feature combinations are extracted from the prediction methods produced by the learning agents. The input representations of the learning agents are then modified by including therein a feature combination extracted from another learning agent. In another part of the method, the parameter values of the learning agents are changed to improve the accuracy of the prediction method. A fitness measure is determined for each learning agent based on the prediction method the agent produces. Parameter values of a learning agent are then selected based on the agent's fitness measure. Variation is introduced into the selected parameter values, and another learning agent of the same class is defined using the varied parameter values. The learning agents are then again trained on the data set to cause a learning agent to produce a prediction method based on the derived feature combinations and varied parameter values.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

SYSTEM AND METHOD FOR COMBINING MULTIPLE LEARNING AGENTS TO PRODUCE A PREDICTION METHOD

FIELD OF THE INVENTION

This invention relates generally to artificial intelligence and machine learning. More particularly, this invention relates to a system and method for combining learning agents that derive prediction methods from a training set of data, such as neural networks, genetic algorithms and decision trees, so as to produce a more accurate prediction method .

BACKGROUND OF THE INVENTION

Prior machine learning systems typically include a database of information known as a training data set from which a learning agent derives a prediction method for solving a problem of interest. This prediction method is then used to predict an event related to the information in the training set. For example, the training set may consist of past information about the weather, and the learning agent may derive a method of forecasting the weather based on the past information. The training set consists of data examples which are defined by features, values for these features, and results. Continuing the example of weather forecasting, an example may includes features such as barometric pressure, temperature and precipitation with their corresponding recorded values (mm of mercury, degrees, rain or not). The resulting weather is also included in the example (e.g., it did or did not rain the next day). The learning agent includes a learning method, a set of parameters for implementing the method, and an input representation that determines how the training data's features will be considered by the method. Typical learning methods include statistical/Bayesian inference, decision-tree induction, neural networks, and genetic algorithms. Each learning method has a set of parameters for which values are chosen for a particular implementation of the method, such as the number of hidden nodes in a neural network. Similarly, each application of a learning method must specify the representation, that is, the features to be considered; for example, the semantics of the neural network's input nodes.

One clear lesson of machine learning research is that problem representation is crucial to the success of all learning methods (see, e.g. Dietterich, T., "Limitations on Inductive Learning," *Proceedings of Sixth International Workshop on Machine Learning* (pp. 125-128), Ithaca, NY: Morgan Kaufman (1989); Rendell, L., & Cho, H., "Empirical Learning as a Function of Concept Character," *Machine Learning*, 5(3), 267-298 (1990); Rendell L., & Ragavan, H., "Improving the Design of Induction Methods by Analyzing Algorithm Functionality and Data-based Concept Complexity," *Proceedings of UCAI*, (pp. 952-958), Chambery, France (1993), which are all hereby incorporated by reference). However, it is generally the case that the choice of problem representation is a task done by a human experimenter, rather than by an automated machine learning system. Also significant in the generalization performance of machine learning systems is the selection of the learning method's parameter values, which is also a task generally accomplished by human "learning engineers" rather than by automated systems themselves.

The effectiveness of input representations and free parameter values are mutually dependent. For example, the appropriate number of hidden nodes for an artificial neural network depends crucially on the number and semantics of the input nodes. Yet up to now, no effective method has been developed for simultaneously searching the spaces of representations and parameter values of a learning agent for the optimum choices, thereby improving the accuracy of its prediction method.

Machine learning systems have also conventionally used a single learning agent for producing a prediction method. Until now, no effective method has been developed that utilizes the interaction of multiple learning agents to produce a more accurate prediction method.

An objective of the invention, therefore, is to provide a method and system that optimizes the selections of parameter values and input representations for a learning agent. Another objective of the invention is to provide a simple yet effective way of synergistically combining multiple learning agents in an integrated and extensible framework to produce a more accurate prediction method. With multiple and diverse learning agents sharing their output, the system is able to generate and exploit synergies between the learning methods and achieve results that can be superior to any of the individual methods acting alone.

SUMMARY OF THE INVENTION

A method of producing a more accurate prediction method for a problem in accordance with the invention comprises the following steps. Training data is provided that is related to a problem for which a prediction method is sought, the training data initially represented as a set of primitive features and their values. At least two learning agents are also provided, the agents including input representations that use the primitive features of the training data. The method then trains the learning agents on the data set, each agent producing in response to the data a prediction method based on the agent's input. Feature combinations are extracted from the prediction methods produced by the learning agents. The input representations of the learning agents are then modified by including feature combinations extracted from another learning agent. The learning agents are again trained on the augmented training data to cause a learning agent to produce another prediction method based on the agent's modified input representation.

In another method of the invention, the parameter values of the learning agents are changed to improve the accuracy of the prediction method. The method includes determining a fitness measure for each learning agent based on the quality of the prediction method the agent produces. Parameter values of a learning agent are then selected based on the agent's fitness measure. Variation is introduced into the selected parameter values, and another learning agent is defined using the varied parameter values. The learning agents are again trained on the data set to cause a learning agent to produce a prediction method based on the varied parameter values.

The two methods may be used separately or in combination. Results indicate a synergistic interaction of learning agents when both methods are combined, which provides yet a more accurate prediction method.

The foregoing and other objects, features, and advantages of the invention will become more apparent from the following detailed description of a preferred embodiment which proceeds with reference to the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of the software architecture of a machine learning system according to the invention.

Fig. 2 is a flowchart of an overall method for producing a prediction method according to the invention.

Fig. 3 is a flowchart of a preferred method of creating an initial agent population according to the overall method of Fig. 2.

Fig. 4 is a flowchart of a preferred method of evaluating learning agents' prediction methods according to the overall method.

Fig. 5 is a flowchart of a preferred method of creating a next population of learning agents according to the overall method.

Fig. 6 is a flowchart of a preferred method of extracting feature combinations according to the overall method.

Fig. 7 is a flowchart of a preferred method of selecting input representations according to the overall method.

Fig. 8 is a pictorial diagram of a population of learning agents created in accordance with the invention.

Fig. 9 is a pictorial diagram of the method of creating a next population of learning agents in accordance with the invention.

Fig. 10 is a pictorial diagram of the method of selecting input representations in accordance with the invention.

Fig. 11 is a graph of the fitness of prediction methods as a function of the generations of learning agents executed.

Fig. 12 is a graph of the contributions of the inventive methods to the accuracy of prediction methods as a function of the generations of learning agents executed.

Fig. 13 is a graph of the accuracy of prediction methods resulting from the synergistic interaction of two learning agents as a function of the generations of learning agents executed.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

The invention can be implemented in any number of ways with a computer program executing on a general purpose computer. The preferred embodiment includes a control program written in Carnegie Mellon Common Lisp 17f and the PCL Common Lisp Object System, resident on a computer-readable medium such as a hard or floppy disk, tape cartridge, etc., and running on a Silicon Graphics Indigo^2 computer workstation. The learning agents incorporated into this embodiment are all

publicly available and include the C4.5 decision tree induction system, the C4.5rules extraction tool (Quinlan, 1991), the LFC++ constructive induction program (Rendell & Ragavan, 1993; Vilalta, 1993), and the conjugate gradient descent trained feedforward neural network (CG) from the UTS neural network simulation package (van Camp, 1994). Each of these learning agents has a method associated with an object class which defines how to execute the method, how to parse the results returned, and what the list, or vector, of free parameters is for the agent. Of course, the invention is not limited to the described computer or learning agents. Any computer system (single CPU, multiprocessor, etc.) capable of executing the control program and learning agents can be used, and any learning agents that produce prediction methods can be used.

Overview of the Machine Learning System

Fig. 1 is a block diagram of the software architecture of a machine learning system 20 according to the invention. A population of learning agents such as agents 24-28 are controlled by an agent control program 30. Each learning agent includes a learning method, a set of parameters for implementing the method, and a representation of training data's features to be considered by the method. Fig. 8 shows a typical agent population comprising four classes of learning agents: statistical/Bayesian inference method-based agents 24a-c, decision-tree induction method-based agents 26a-d, neural network method-based agents 28a-c, and genetic algorithm method-based agents 29a-c. Each learning agent includes a set of parameters values, such as the number of hidden nodes in a neural network (five hidden in agent 28a, zero hidden in agent 28b). Similarly, each learning agent includes a representation of the features considered by the learning method, such as the semantics of the neural network's input nodes (F1, F2/F3, etc.). Control program 30 sets parameter values and executes the learning method associated with each agent. The data input to each learning agent is derived from a knowledge data base such as training data set 32 by transforming the original data set according to a set of feature combinations specific to each agent. These specific transformation methods are represented in Fig. 1 by transforms 35, 36, and 37. However, in the initial execution of system 20, all agents are given the original, untransformed version of the training data.

Training data set 32 includes data examples which are defined by features, values for these features, and results. If the training set relates to weather prediction, an example may includes features such as barometric pressure, temperature and precipitation with their corresponding recorded values (mm of mercury, degrees, rain or not). The resulting weather is also included in the example (e.g., it did rain the next day or not).

Each of the learning agents 24-28 produces in response a prediction method based on the agent's input representation. The prediction methods are examined by a prediction method evaluator 38 that determines an unbiased estimate of the accuracy of the prediction method generated by each agent. The prediction accuracies are used by control program 30 to select new parameter settings for a next execution of the learning agents.

The prediction methods produced by the learning methods are also examined by a compound feature constructor 39. Constructor 39 extracts the combination of features in the agent's

input representation that are determined to be the most important in producing the agent's prediction method. These extracted feature combinations are passed to the appropriate data transforms 35-37 for re-representing the training data for the next execution of the learning agents. As will be described, a feature combination can be passed to the same learning agent from whose prediction method it was extracted, to a different agent of the same class, or to a learning agent of a different class.

Both the parameter values and the input representations for learning agents 24-28 change for each system execution, thereby creating a new generation of learning agents each time. Execution of the learning agents continues until a prediction method that achieves a desired accuracy (such as 100%) is found or until a predefined time limit has expired.

The Overall Method

A method for producing a prediction method according to the invention functions by evolving the population of learning agents 24-28. The population is defined by a classification task T, a set of classified examples of that task expressed in a primitive representation E_p, a fitness function for agents f_A, and a set of learning agents A:

$$P_{coev} = \{T, E_p, f_A, A\}$$

A fitness function for agents maps a member of the set A to a real number between 0 and 1. For convenience, it is useful to define P_L to be the subset of P_{coev} where all the agents use learning method L (see below).

Each learning agent A is defined by a learning method L, a vector of parameter values v, a fitness function for memes f_m, and an ordered set of problem representation transformations R:

$$A_i = \{L, v, f_m, R\}$$

The set, or vector, of parameter values may have different length for different learning methods. Each member of the set of problem representation transformations R_i is a mapping from an example ($e_p \in E_p$) to a value which is a legal element of an input representation for L (e.g. a real number, nominal value, bit, horn clause, etc.). The individual mappings are called *memes*. The ordered set of feature combinations (R_i) is called the agent's *memome*, and the vector of parameter values is called its *genome*. The fitness function for memes f_m is a function that maps a member of the set R_i to a real number between 0 and 1.

A transformed example e_i is the result of sequentially applying each of the problem representation transformations $r_i \in R_i$ to an example $e_p \in E_p$. The application of the set of problem representation transformations to the set of examples in primitive representation results in a set of transformed examples, which is called E_i. When $E_i = E_p$, the transformation is said to be the identity.

Given a population as defined above, the process of coevolution is shown in Fig. 2 and in the pseudocode in Table 1.

Initialize the population with random legal parameter vectors and the identity representation transformation.
* Determine the phenotype of each agent i by applying learning algorithm L_i to transformed examples E_i using parameter values v_i using K-way cross-validation. The phenotype is an ordered set of: the output of the learning agent's cross-validation runs, the cross validation accuracies and the learning time.
Determine the fitness of each agent by applying f_A to the phenotype of each agent.
Create new memes by parsing the output in each learning agent's phenotype and extracting important feature combinations. The union of all memes generated by all members of the population is called the meme pool.
Exchange memes. For each agent i, apply its meme fitness function F_m to elements of the meme pool. Select the agent's target number of memes (a free parameter) from the meme pool with a probability proportional to the fitness of the memes.
Repeat from * meme-transfers-per-generation times
Determine phenotype of the agents with their new memes
Determine the fitness of the agents
Create the next generation of agents by mutation, crossover and fitness proportional reproduction. Agents whose genomes are mutated keep their memomes intact. For each pair of agents whose genomes are crossed over to create a new pair of agents, the memomes of the parents are arbitrarily assigned to the offspring.
Repeat from * until a member of the population can solve the problem

Table 1

As a first step 40, an initial population of learning agents is created. These agents are typically though not necessarily of different classes, such as agents 24-28. Each of the learning agents is then trained on the training data (step 42). This training comprises supplying the data to the learning methods in a form that allows the methods to utilized their parameter values and input representations. As a result of the training, each of the learning agents produces a prediction method. The prediction methods are evaluated to determine their fitness (step 44). A decision based on the fitness measure (or time expired) is then made to select a prediction method or to continue to evolve the learning agents (step 46). If time has expired or the fitness measure of a prediction method is sufficient, then the method ends. If neither of these events occurs, then the method proceeds to produce another set of prediction methods.

The next step in the method is to extract feature combinations from the trained learning agents for use in future input representations (step 50). With knowledge of these extracted feature combinations, the control program transforms the features of the training data to reflect the future input representations (step 52). For example, if it determined that the mathematical combination of F1 times F2 is important for the learning agents, then the training data is transformed to provide a feature that has the value of F1 times F2.

Another step in the method is to select parameter values from the learning agents based on the fitness of their prediction methods and create therefrom a next population of learning agents (step 54). As will be described, this selection includes copying and varying parameter values among learning agents in a class.

The method concludes with selecting input representations for the next population of learning agents (step 56). This step is the process for evaluating the various fitness combinations extracted from the prediction methods and then matching them with the different learning agents in the population.

Creating the Initial Agent Population

Fig. 3 is a flowchart of a preferred way of creating an initial agent population according to the invention. To begin, each learning agent has a set of primitive features as its initial input representation (60). There are at least two classes of learning agent, each with a list of parameters and their legal ranges (62). Training data set 22 is also provided (64).

First, a learning agent is selected from a class, such as a neural network-based agent 28a (step 60). The agent's parameter values are then selected at random within their legal ranges (step 62). The agent's input representation is also set to an initial representation (step 64). A decision is then made if there are enough learning agents provided (step 66). If not, more agents of various classes are created (steps 60-66). If so, the creating step is complete, and the method proceeds with training the agents population (step 68).

Evaluating Agents' Prediction Methods

In addition to defining the fitness of features, system 20 must define the fitness of learning agents themselves for the evolution of parameter values.

Fig. 4 is a flowchart of a preferred way of evaluating the agents' prediction methods. While a number of techniques are known in the art for evaluating the accuracy of a prediction method, the preferred way uses the following steps. First, cross validation is applied to the prediction method of each learning agent to determine an unbiased estimate of the method's accuracy (as a percent of correct predictions) (step 70). Cross validation is a well established statistical technique that is widely know in the machine learning art. See, for example, its description in *Computer Systems That Learn*, by Weiss and Kulikowski (1991). Additionally, the time required for a learning agent to produce its prediction method is determined (step 72). The cross validation number and time are then combined to calculate a fitness measure of the trained agent (step 74).

The fitness function for learning agents was selected to evolve learning agents that are accurate, robust and fast:

$$f(A_i) = C(A_i) - \sqrt{S(A_i)} - k\left(\frac{t_{A_i} - t_A}{S(t_A)}\right)$$

where $C(A_i)$ is the cross-validation accuracy of the agent A_i, $S(A_i)$ is the standard deviation of that accuracy, t_{Ai} is the time it took for that agent to learn, t_A is the mean execution time for that type of agent, $S(t_A)$ is the standard deviation of that mean, and k is a constant that trades off the value of accuracy versus that of learning time. Execution time is measured as the number of standard deviations from the mean learning time for that class of learning agents so that classes of learning agents with different training times can coexist in the same population. In the inventive system, k was selected to be 3 and accuracies and standard deviations are measured in percentage points. So, a learning agent that had a mean accuracy of 80%, a standard deviation of 9% and took 1 standard deviation less than the mean training time for that class of learning agent would get a fitness score of $80 - \sqrt{9}-(-1)=78$.

Extracting Feature Combinations (Memes)

The main difference between coevolution learning and other evolutionary methods derives from the creation and exchange of memes. The meme creation process takes the output of learning agents that have been applied to a particular problem, and identifies combinations of the input features that the learning agent determined were relevant in making the desired distinction. The process of parsing output and creating new memes is specific to each learning method. For example, a program that learns rules from examples might create new memes from the left hand sides of each of the induced rules. Or, a program that learned weights in a neural network might create new memes that were the weighted sum of the inputs to each of its hidden nodes (perhaps thresholded to remove marginal contributions). Specific meme generation methods are discussed in more detail in the implementation section, below.

Fig. 5 is a flowchart of a preferred way of extracting feature combinations from trained learning agents. Different classes of agents require different extraction technique, and the extraction technique selected is based on the agent class (step 80). For genetic algorithm-based agents 30, the algorithms with the highest accuracy are selected (step 82) and are converted to rules (step 84). For statistical/Bayesian inference method-based agents 24a-c, the statistics with the highest correlation are selected (step 86) and converted to rules (step 88). For neural network method-based agents 28, the rules are extracted from the network (step 90) using known techniques such as described by Craven and Shavlik in "Extracting Tree-structured Representations of Trained Network," *Proceedings of the Conference on Neural Information Processing Systems* (1996) or by Towell and Shavlik in "The Extraction of Refined Rules form Knowledge-based Neural Network." *Machine Learning* (1993). For decision-tree induction method-based agents 26, the rules are extracted from the tree (step 92) as done, for example, by the program C4.5rules described above and incorporated herein by reference. The left hand sides of the rules, from whatever agent, are then used to define the feature combinations.

Creating a Next Population of Learning Agents

The creation of the next generation of agents is a minor variant on traditional genetic algorithms. Instead of having the genome be "bits" it is a vector of parameter values. The crossover, mutation and fitness proportional reproduction functions are all identical with the related operations on bit vectors in genetic algorithms. It is, of course, possible to transform parameter vectors into bitstring

representations themselves, if it were desirable. In addition to using parameter vectors instead of bits, the other difference is that each subpopulation using a particular learning method (the P_L's) has its own type of parameter vector, since the free parameters and their legal values vary among learning methods. These parameter vectors may be of different sizes, so crossover can only be applied with members of the same subpopulation.

Fig. 6 is a flowchart of a preferred method for creating a next population, or generation, of learning agents in accordance with the invention. The steps of this method are best understood with reference to Fig. 9, which shows how they apply to change the parameter values of the three neural network-based agents 28a-c. From the evaluation of the agents' prediction methods(Fig. 3), fitness measures for each of the agents have been calculated. These are shown in Fig. 9 as decimal fractions such as .67574 for agent 28a, .6823 for agent 28b, and .1867 for agent 28c. Parameter values for each agent in the next generation are then set (step 100). These are set preferably by randomly selecting a trained agent within the class (e.g., neural network-based agents) and copying its parameter values, with the probability of selection proportional to the fitness measure of the trained agent. For example, in Fig. 9, because of the relative fitness measures agent 28a is selected once for the next generation of agents (agent 28a'), agent 28b is selected twice (agents 28b' and 28b"), and agent 28c is not selected.

In a next step 102, a subset of the new agents is then randomly selected. In Fig. 9, this subset is agent 28b'. The subset, of course could be more than one new agent. For each agent of the subset, a parameter is randomly selected, and its value is randomly set (step 104). In the example of Fig. 9, the parameter indicating the number of hidden nodes in the neural network is selected and its value is changed from 3 to 1. This creates a mutated, final agent version 28f that differs from the intermediate form 28b' by the number of hidden nodes.

One or more pairs of agents are then selected from the intermediate population (step 106) and parameter values are crossed over, or swapped (step 108). In the example of Fig. 9, agents 28a' and 28b" are the pair of selected agents, and the parameter values for back propagation and lambda are swapped to form two new agents 28d and 28e.

The next population of learning agents is then ready for training on the training data. In Fig. 9 , this next population comprises agents 28 d-f in the neural network class. Other classes of learning agents would similarly have new generations.

Selecting Input representations for Learning Agents

Memes (feature combinations) must be assigned fitnesses by agents. It is possible for each agent to have its own method of determining meme fitness. However, a simplified method is used in the preferred embodiment of the invention. Whenever a program generates a new meme, its fitness is defined to be the average "importance" of the feature combination, weighted by the accuracy of the learning agents that used the meme. In Durham's terms, the inventor of the new meme "imposes" its belief in its value on others. In addition, receivers of memes have a parameter which determines how much weight they put on memes they generate themselves versus memes generated by others. This

mechanism could be straightforwardly generalized to weight the fitnesses of memes generated by different classes of agents differently. Importance is defined differently for different learning methods. For C4.5 and LFC++, the importance of a feature combination is the ratio of correctly classified examples that triggered the rule containing the feature to the total number of occurrences of the feature. So, for example, if "A and not B" is a feature extracted from a single C4.5 learning agent that had a 80% cross-validation accuracy, and 9 of the 10 examples the feature appeared in were classified correctly by the rule containing the feature, it's importance would be 9/10 and its fitness would be 0.9 * 0.8 = 0.72. For UTS, the importance of a feature is the absolute value of the ratio of the weight from the hidden node that defined the feature to the threshold of the output unit. If there is more than one output unit, it is the maximum ratio for any of the output units. Features that play a significant role in learning agents that are accurate have higher fitness than features that do not play as much of a role or appear in learning agents that are less accurate. It is possible for a feature to have relatively low prevalence in the population and yet still be important if it tends to generate correct answers when it is used.

Fig. 7 is a flowchart that illustrated the process for selecting input representations for learning agents. First the fitness for each feature (i.e., feature combination) is calculated (step 110) in the manner described above. Then for each agent, the number of features specified by its "feature count: parameter is selected (step 112). A decision is then made whether to select a feature generated by an agent of the same class (step 114). If so, the feature is selected based on its fitness (step 116). If not, a feature from another class of agents is selected, again based on its fitness (step 118). A further decision is then made whether the learning agent has selected enough features for its input representation (step 120). If not, then steps 114-120 are repeated until enough features have been selected (step 122).

The process of Fig. 7 is shown graphically in Fig. 10. Learning agents such as agents 28a, 26c, 29a, and 24b each produce a prediction method from which feature combinations are extracted by constructor 39 (Fig. 1) and given a fitness measure before being collected in a feature combination pool 122. From this pool feature combinations are then selected for the next generation of learning agents such as 28d, 26f, 29e, and 24f.

A Specific Implementation

As described, a current implementation of the invention includes the C4.5 decision tree induction system and C4.5rules rule extraction tool, the LFC++ constructive induction program and the conjugate gradient descent trained feedforward neural network (CG) from the UTS neural network simulation package. Most of the documented parameters for each of class of learning agent is included in the free parameter vector for that system, along with a specification of either an upper and lower bound for the parameter's values or a list of possible parameter values. For example, the parameter vector for the UTS conjugate gradient descent learning agent includes the number of hidden nodes in the network, the maximum number of iterations before halting, the output tolerance (specifying how close to 1 or 0 an output has to be to count as true or false), a flag for whether or not to

use competitive learning on the output, two nominal parameters specifying the kind of line search and direction finding method to use, and three parameters specifying the maximum number of function evaluations per iteration, the minimum function reduction necessary to continue the search and the maximum slope ratio to continue the search. Theses parameters are explained more fully in the documentation available with the source code.

Each learning agent has a method for extracting new features from its output. LFC++, like other constructive induction programs, specifically defines new features as part of its output. For C4.5, the output of the C4.5rules tool was parsed so that each left hand side of each rule was reified into a new feature definition. For the neural network learning agent, a new feature was created for each hidden node defined by the weighted sum of the inputs to that node, with any input whose contribution to that sum was less than the threshold of the node divided by the number of inputs removed from the sum. These extracted features are added to the feature (i.e., meme) pool.

Several other aspects of this implementation must be specified in order for it to work properly. Meme definitions must be stored in a canonical form so that it is possible to detect when two or more generated memes are in effect identical and should have their fitness scores combined. Memes are represented as boolean combinations of primitives or mathematical formula over primitives, which can be straightforwardly compared, although the extension to first order predicate calculus would make this a more difficult problem. The number of memes that an agent uses (i.e. the dimensionality of its input representation) is treated as a free parameter of the agent, and allowed to vary from two to twice the number of primitives. In addition, agents are allowed to prefer memes of their own creation to memes created by other learning agents. A single parameter, ranging over [0,1] specifies the internal meme preference of each agent.

Simulation Results on a Simple Test Problem

The implementation of system 20 described above was applied to an artificial problem designed to be moderately difficult for the machine learning programs included in the system. The problem was to identify whether any three consecutive bits in a nine bit string were on, e.g. 011011011 is false, and 001110011 is true. This problem is analogous both to detecting a win in tic-tac-toe and to solving parity problems, which are known to be difficult for many types of learning systems. There are 512 possible examples, and the problem is simple enough so that the machine learning systems used (even the neural network) run reasonably quickly. This is important, since each program is executed a large number of times (see the computational complexity section, below).

This problem was run on a system that used only LFC++ and the CG learning agents. The population consisted of 10 LFC++ learning agents and 10 CG learning agents, and was run for 10 generations. Six fold cross-validation was used, and there was one meme exchange per generation. This rather modest problem therefore required 2400 learning agent executions, taking more than four days of R4400 CPU time. (The CPU time use was dominated by the 1200 neural network training runs.) Despite the large amount of computation required for this simple problem, it is reasonable to

hope that the amount of computation required will grow slowly with more difficult problems (see the computational complexity section, below).

This simulation illustrated three significant points. First, the implementation of system 20 was able to consistently improve as it evolved. The improvement was apparent in both the maximum and the average performance in the population, and in both the cross-validation accuracies of the learning agents and in their fitnesses (i.e. accuracy, speed and robustness). Figure 11 shows these results. In ten generations, the average fitness of the population climbed from 51%% to 75%, and the standard deviation fell from 21% to 12.5%. Looking at the best individual in the population shows that the maximum fitness climbed from 84% to 94%, and the maximum cross-validation accuracy climbed from 82.3% to 95%. Average execution time fell by more than 25% for both classes of learning agents (it fell somewhat more sharply, and in much greater absolute magnitude for the neural network learning agents.)

Figure 12 illustrates the separate contributions of the genetic evolution component and the memetic evolution component compared to coevolution in the performance of the best individual learning agent in the population. Consider first the purely genetic evolution of the free parameter vectors. When there is no memetic component, there is no interaction between the different classes of learning agents, since crossover and other sources of genetic variation apply only to a specific type of learning agent. Using genetic search to find the best free parameter values for a neural network is known to be slow (Yao, 1993). In this genetic-only neural network simulation, only a small improvement was found in the fifth generation. Similarly for the genetic evolution of LFC++'s free parameters, a modest difference was found in the sixth generation. Memetic evolution alone (without changing any of the free parameter values) showed a more significant effect. Because both the neural network and the constructive induction program make contributions to memetic evolution, any synergies between them will appear in the memetic-only curve (see discussion of figure 4, below). However, the steady rise of the coevolution curve, compared to the memetic-evolution only curve suggests that some adjustment of the free parameter values to match the representation changes induced by memetic transfer may have a positive effect. At the end of 10 generations, the maximum accuracy of the coevolved population is more than 5 percentage points higher than the memetic only population. However, it is worth noting that the memetic only population equaled that performance figure earlier in the simulation, and then drifted away.

Figure 13 illustrates the synergy that coevolution finds between the CG agent population and the LFC++ agent population. When run with only a single type of learning agent, the memetic evolution part of the inventive system becomes a kind of constructive induction program, where the output of the learning agent is feed back as an input feature in the next iteration. Since LFC++ is already a constructive induction program, it is somewhat surprising to note that LFC++ agents coevolving with each other still manage a small amount of improvement. Perhaps this is due to the fact that the appropriate features for this problem have three conjuncts, and LFC++ under most free parameter values tends to construct new features from pairs of input features. CG alone does not do

very well on this problem. Even when coevolved with itself, its most accurate agent improves only from about 60% to a bit over 70% in ten generations. However, when these two learning methods are combined in a single coevolution learning agent, the results are clearly better than either of the methods used alone. The gap between the best performance of the pair of methods and the best single method tends to be about 5 percentage points in this simulation, and the difference tends to be larger as the population evolves.

In view of the many possible embodiments to which the principles of the invention may be applied, it should be recognized that the illustrated embodiment is only a preferred example of the invention and not as a limitation on its scope. The scope of the invention, rather, is defined by the following claims. I therefore claim as my invention all that comes within the scope and spirit of these claims.

I claim:

1. A computer-implemented method of producing a prediction method for a problem, the method comprising the following steps:

providing training data related to a problem for which a prediction method is sought, the training data represented by features with values;

providing at least two learning agents, the agents including input representations;

training the learning agents on the data set, each agent producing in response to the data a prediction method based on the agent's input representation;

extracting feature combinations from the prediction methods produced by the learning agents;

modifying the input representation of a learning agent by including in the input representation a feature combination extracted from another learning agent; and

again training the learning agents on the data set to cause a learning agent to produce a prediction method based on the agent's modified input representation.

2. The method of claim 1 wherein the extracting step comprises:

analyzing a prediction method to identify which features of an agent's input representation are more important and less important to the prediction method; and

combining at least two of the more important features to produce a feature combination.

3. The method of claim 1 including:

collecting feature combinations extracted from the prediction methods; and

determining a fitness measure for each of the collected feature combinations,

wherein the modifying step includes selecting a feature combination for an agent's input representation based on the feature combination's fitness measure.

4. The method of claim 3 wherein the step of determining a fitness measure for a feature combination comprises:

globally assigning a fitness value to each feature combination;

combining the combination's fitness value with a preference value to produce the fitness measure.

5. The method of claim 1 wherein the learning agents are of two different types.

6. The method of claim 1 wherein the learning agents are of the same type but have different parameter values.

7. The method of claim 1 wherein the learning agents have a same list of parameters but different parameter values, the method including:

determining a fitness measure for each learning agent based on the prediction method the agent produces;

selecting the parameter values of a learning agent based on the agent's fitness measure;

introducing variation into the selected parameter values; and

defining another learning agent using the varied parameter values.

8. The method of claim 7 wherein the step of determining a fitness measure comprises determining an accuracy of the prediction method and a completion time for the learning agent.

9. A computer-readable medium on which is stored a computer program comprising instructions which, when executed, perform the steps of claim 1.

10. A computer-implemented method of producing a prediction method, the method comprising the following steps:

providing training data related to a problem for which a prediction method is sought, the training data represented by features with values;

providing at least two learning agents having a same parameter list but different parameter values;

training the learning agents on the data set, each agent producing in response to the data a prediction method based on the agent's parameter values;

determining a fitness measure for each learning agent based on the prediction method the agent produces;

selecting the parameter values of a learning agent based on the agent's fitness measure;

introducing variation into the selected parameter values;

defining another learning agent using the varied parameter values; and

again training the learning agents on the data set to cause a learning agent to produce a prediction method based on the varied parameter values.

11. A computer-implemented method of producing a prediction method, the method comprising the following steps:

providing training data related to a problem for which a prediction method is sought, the training data represented by features with values;

providing a plurality of learning agents of which at least two have a same parameter list but different parameter values, the learning agents including input representations;

training the learning agents on the data set, each agent producing in response to the data a prediction method based on the agent's parameter values and input representation;

determining a fitness measure for each learning agent based on the prediction method the agent produces;

selecting the parameter values of a learning agent based on the agent's fitness measure;

introducing variation into the selected parameter values;

defining another learning agent using the varied parameter values;

extracting feature combinations from the prediction methods produced by the learning agents;

modifying the input representation of a learning agent by including in the input representation a feature combination extracted from another learning agent; and

again training the learning agents on the data set to cause a learning agent to produce a prediction method based on the varied parameter set and the modified input representation.

12. A computer-implemented system for producing a prediction method for a problem, comprising:

training data related to a problem for which a prediction method is sought, the training data represented by features with values;

at least two learning agents, the agents including input representations; and

at least one computer programmed for:

training the learning agents on the data set, each agent producing in response to the data a prediction method based on the agent's input representation;

extracting feature combinations from the prediction methods produced by the learning agents;

modifying the input representation of a learning agent by including in the input representation a feature combination extracted from another learning agent; and

again training the learning agents on the data set to cause a learning agent to produce a prediction method based on the agent's modified input representation.

13. The system of claim 12 wherein at least two learning agent have a same parameter list but different parameter values, and the computer is further programmed for:

training the learning agents on the data set, each agent producing in response to the data a prediction method based on the agent's parameter values;

determining a fitness measure for each learning agent based on the prediction method the agent produces;

selecting the parameter values of a learning agent based on the agent's fitness measure;

introducing variation into the selected parameter values;

defining another learning agent using the varied parameter values; and

again training the learning agents on the data set to cause a learning agent to produce a prediction method based on the varied parameter values.

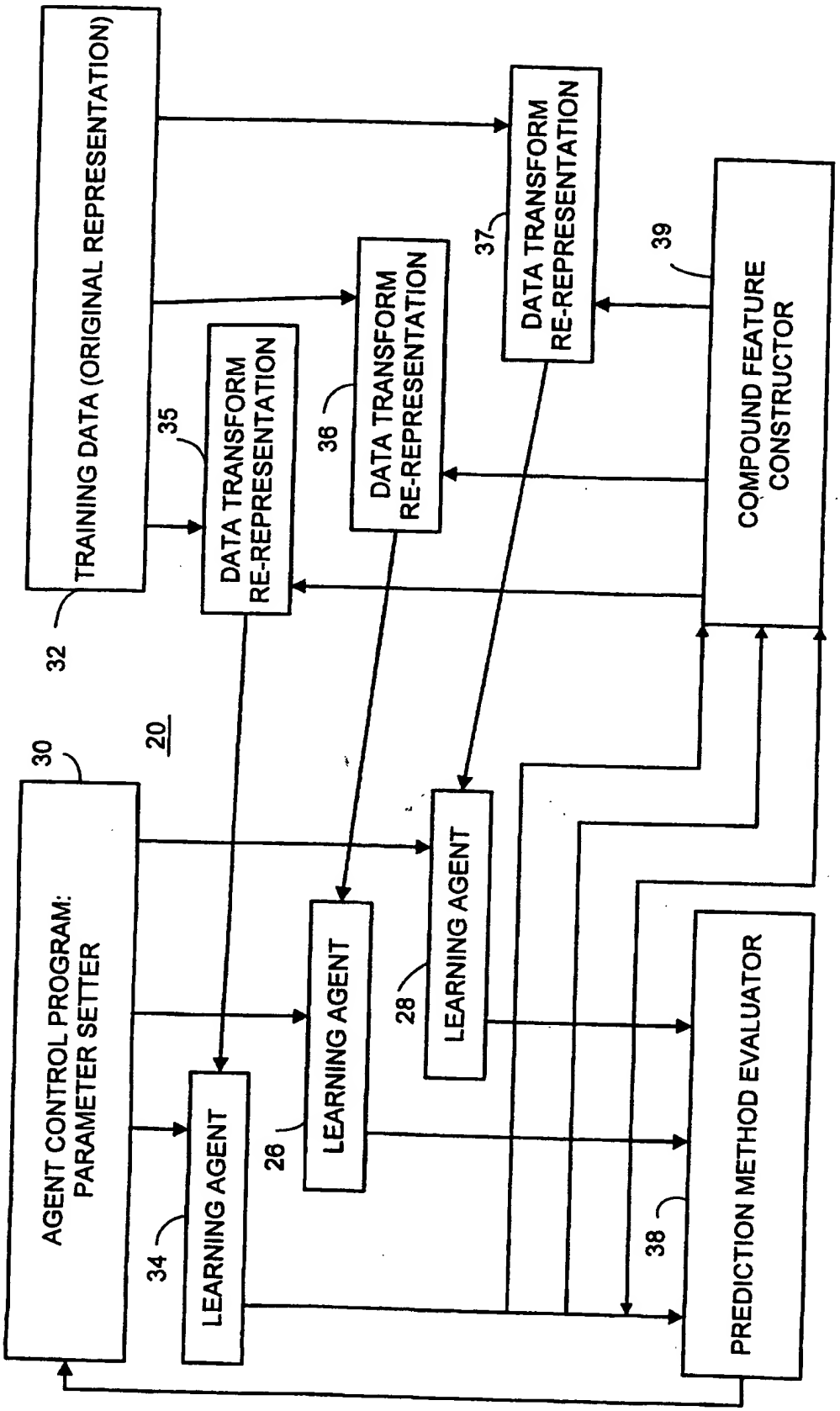
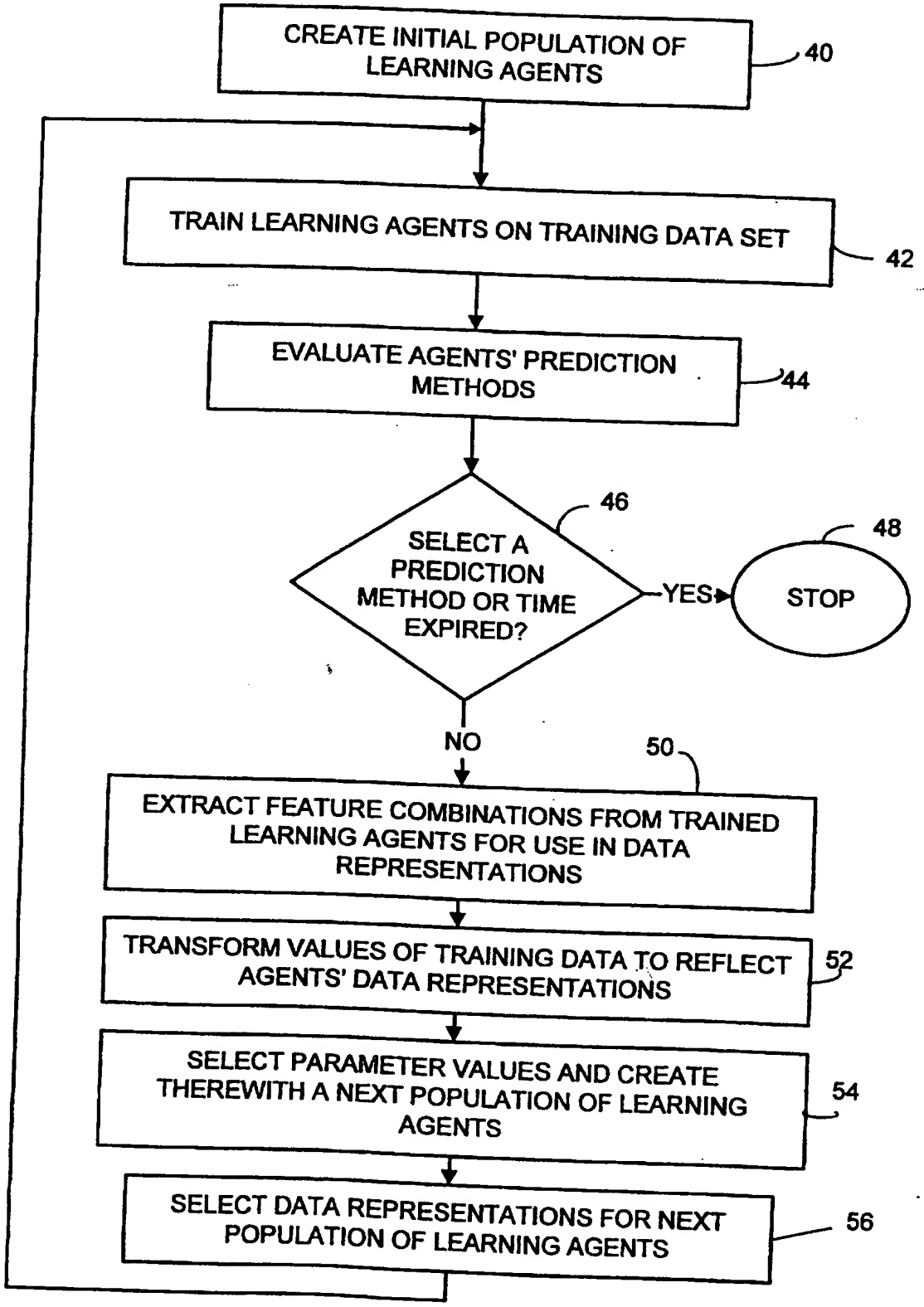

TRAINING DATA (ORIGINAL REPRESENTATION)
32
35
DATA TRANSFORM RE-REPRESENTATION
36
DATA TRANSFORM RE-REPRESENTATION
37
DATA TRANSFORM RE-REPRESENTATION
39
COMPOUND FEATURE CONSTRUCTOR
20
30
AGENT CONTROL PROGRAM: PARAMETER SETTER
34
LEARNING AGENT
26
LEARNING AGENT
28
LEARNING AGENT
38
PREDICTION METHOD EVALUATOR

FIG. 1

FIG. 2


CREATE INITIAL POPULATION OF LEARNING AGENTS
40
TRAIN LEARNING AGENTS ON TRAINING DATA SET
42
EVALUATE AGENTS' PREDICTION METHODS
44
46
SELECT A PREDICTION METHOD OR TIME EXPIRED?
YES
48
STOP
NO
50
EXTRACT FEATURE COMBINATIONS FROM TRAINED LEARNING AGENTS FOR USE IN DATA REPRESENTATIONS
TRANSFORM VALUES OF TRAINING DATA TO REFLECT AGENTS' DATA REPRESENTATIONS
52
SELECT PARAMETER VALUES AND CREATE THEREWITH A NEXT POPULATION OF LEARNING AGENTS
54
SELECT DATA REPRESENTATIONS FOR NEXT POPULATION OF LEARNING AGENTS
56

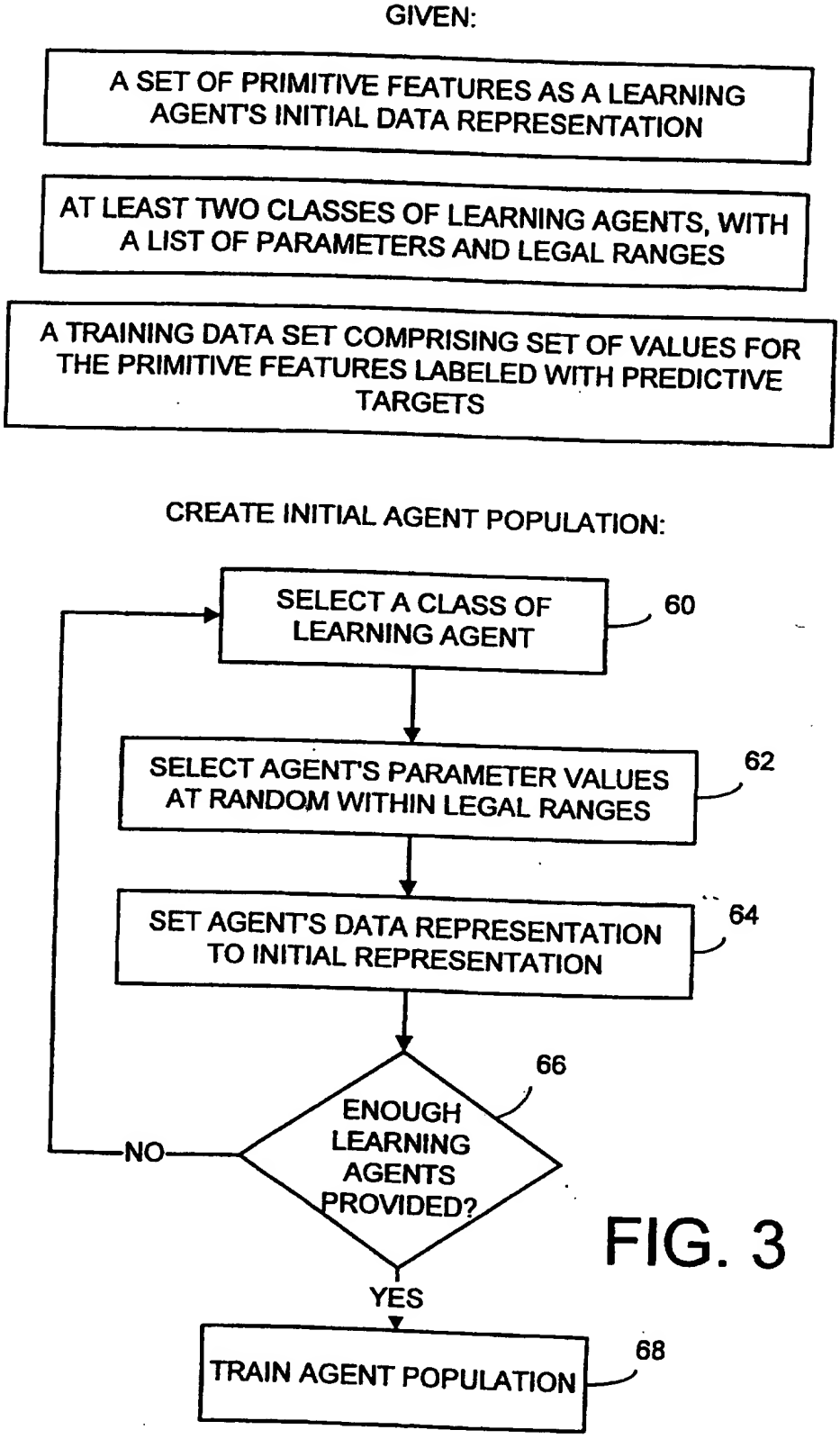

GIVEN:
A SET OF PRIMITIVE FEATURES AS A LEARNING AGENT'S INITIAL DATA REPRESENTATION
AT LEAST TWO CLASSES OF LEARNING AGENTS, WITH A LIST OF PARAMETERS AND LEGAL RANGES
A TRAINING DATA SET COMPRISING SET OF VALUES FOR THE PRIMITIVE FEATURES LABELED WITH PREDICTIVE TARGETS
CREATE INITIAL AGENT POPULATION:
SELECT A CLASS OF LEARNING AGENT
60
SELECT AGENT'S PARAMETER VALUES AT RANDOM WITHIN LEGAL RANGES
62
SET AGENT'S DATA REPRESENTATION TO INITIAL REPRESENTATION
64
ENOUGH LEARNING AGENTS PROVIDED?
66
NO
YES
TRAIN AGENT POPULATION
68

FIG. 3

EVALUATE AGENTS' PREDICTION METHODS:

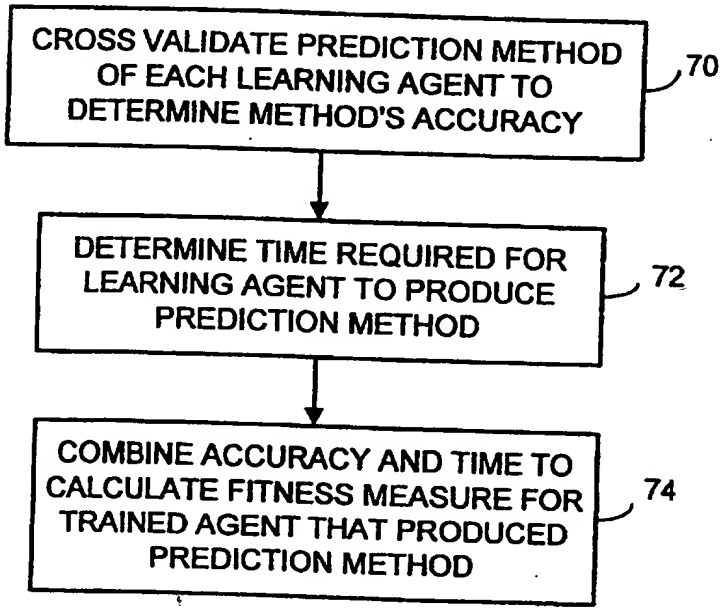


FIG. 4

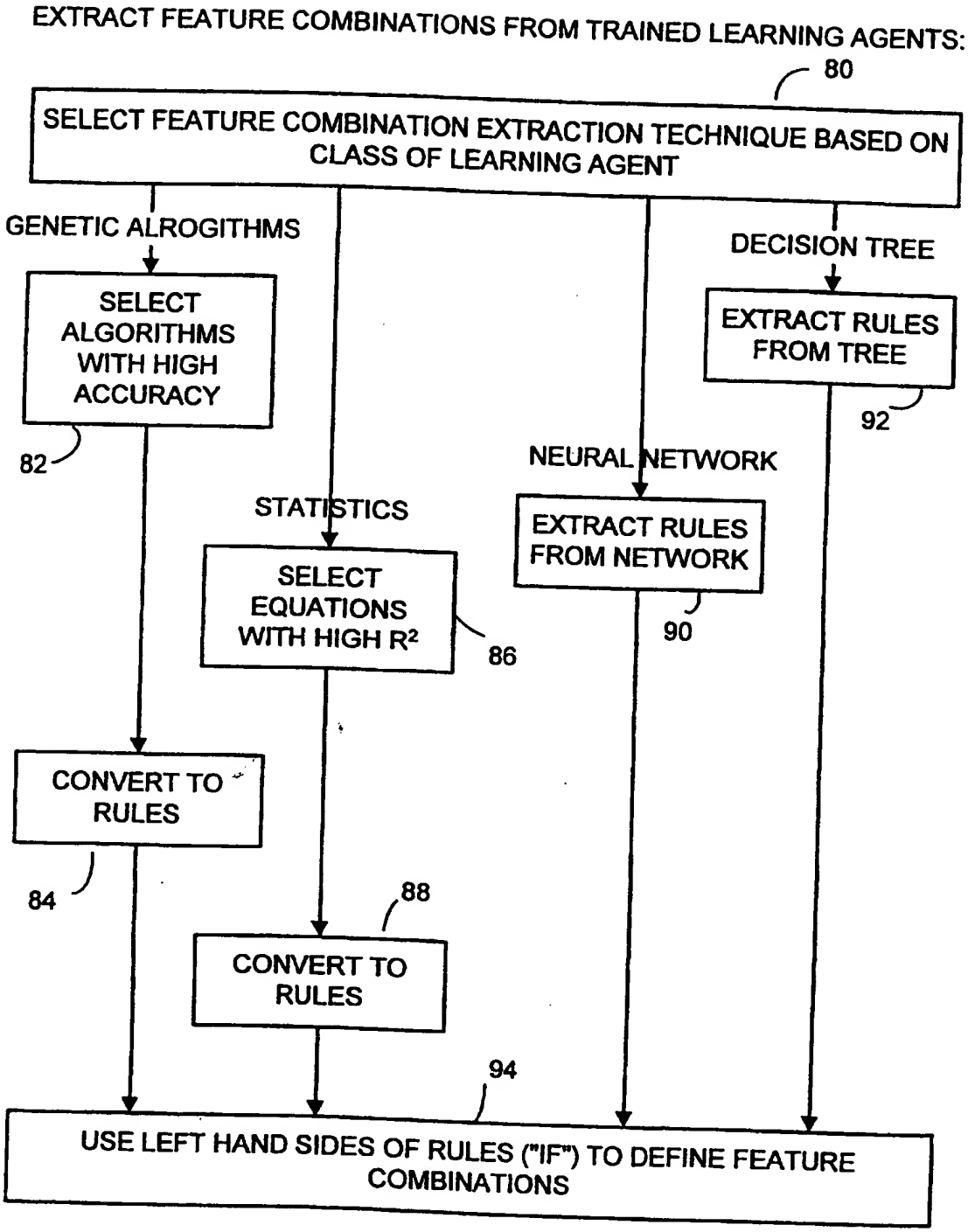

EXTRACT FEATURE COMBINATIONS FROM TRAINED LEARNING AGENTS:
80
SELECT FEATURE COMBINATION EXTRACTION TECHNIQUE BASED ON CLASS OF LEARNING AGENT
GENETIC ALROGITHMS
SELECT ALGORITHMS WITH HIGH ACCURACY
82
STATISTICS
SELECT EQUATIONS WITH HIGH R^2
86
NEURAL NETWORK
EXTRACT RULES FROM NETWORK
90
DECISION TREE
EXTRACT RULES FROM TREE
92
CONVERT TO RULES
84
88
CONVERT TO RULES
94
USE LEFT HAND SIDES OF RULES ("IF") TO DEFINE FEATURE COMBINATIONS

FIG. 5

CREATE A NEXT POPULATION OF LEARNING AGENTS:

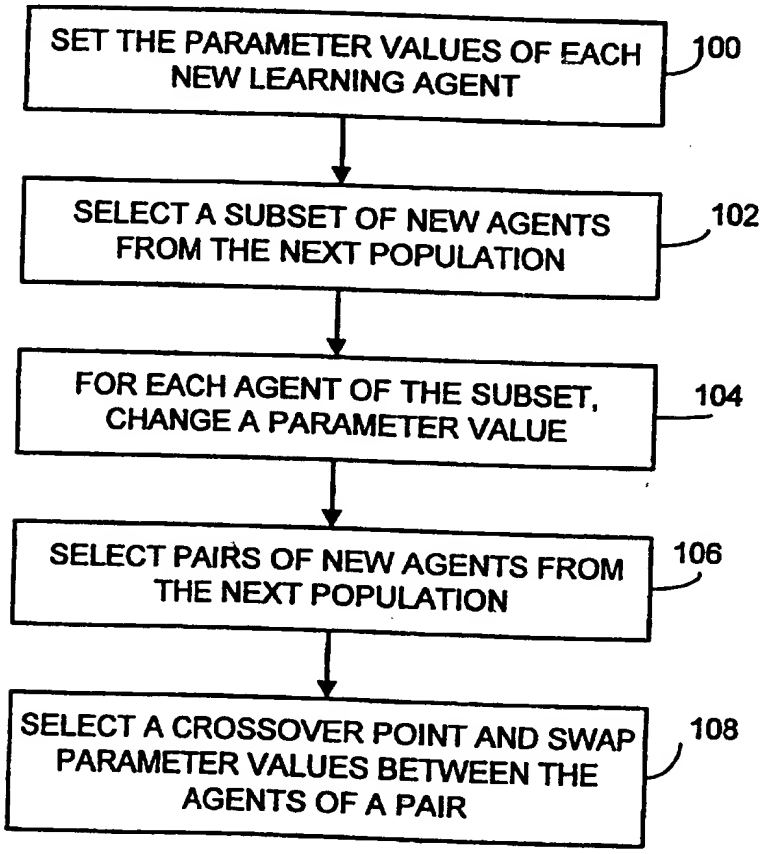


FIG. 6

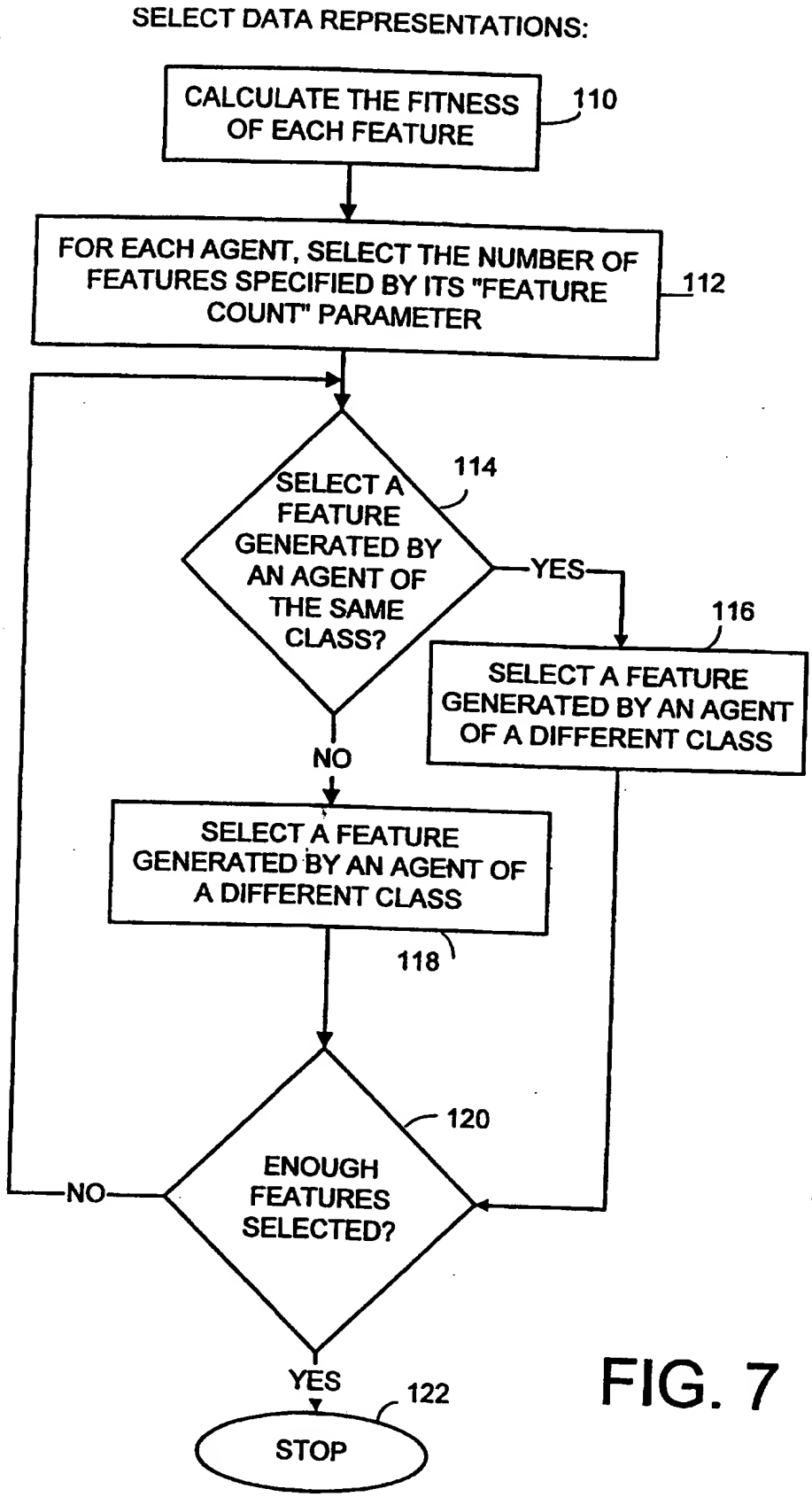

SELECT DATA REPRESENTATIONS:
CALCULATE THE FITNESS OF EACH FEATURE
110
FOR EACH AGENT, SELECT THE NUMBER OF FEATURES SPECIFIED BY ITS "FEATURE COUNT" PARAMETER
112
SELECT A FEATURE GENERATED BY AN AGENT OF THE SAME CLASS?
114
YES
116
SELECT A FEATURE GENERATED BY AN AGENT OF A DIFFERENT CLASS
NO
SELECT A FEATURE GENERATED BY AN AGENT OF A DIFFERENT CLASS
118
ENOUGH FEATURES SELECTED?
120
NO
YES
122
STOP

FIG. 7

FIG. 8

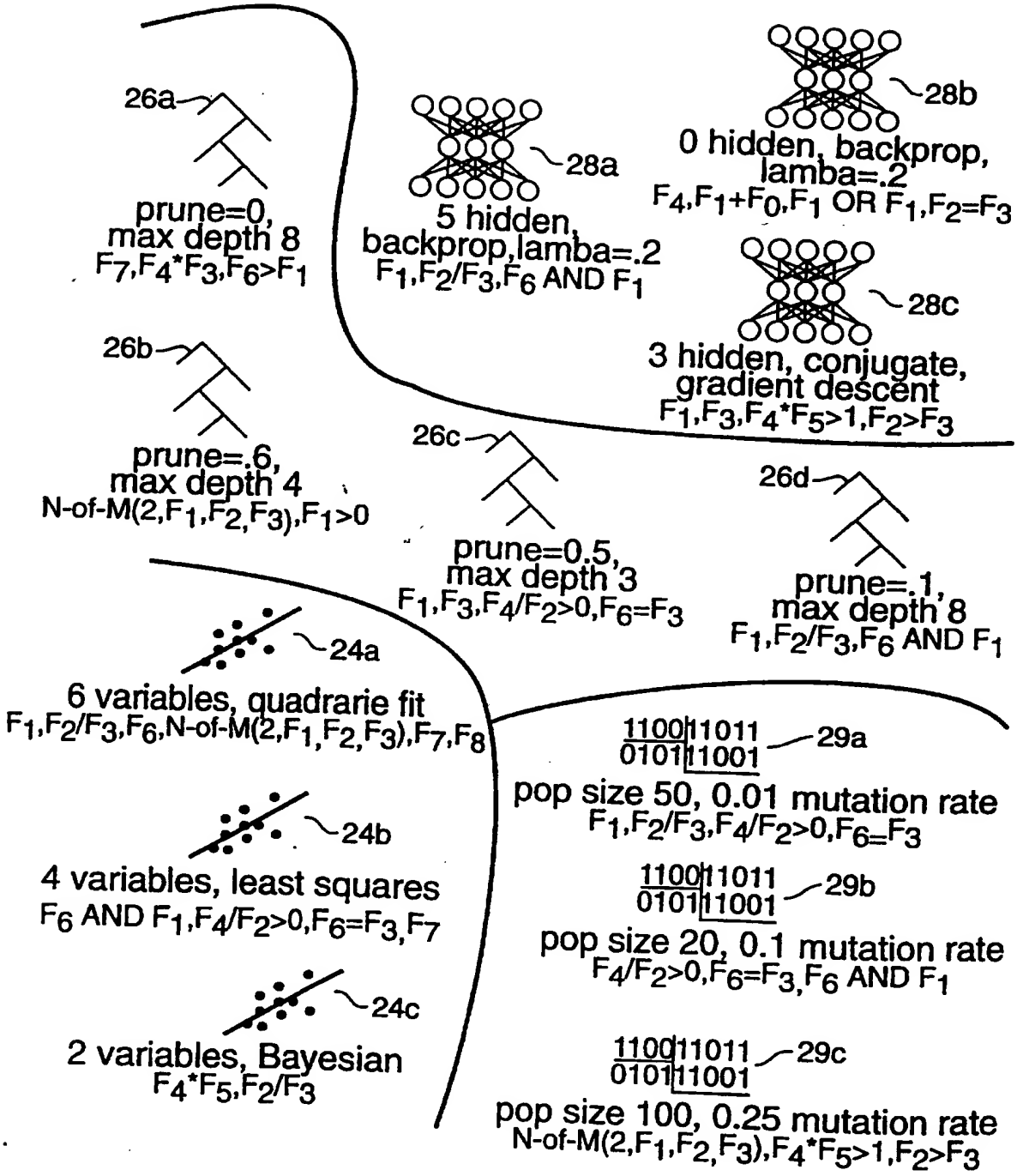
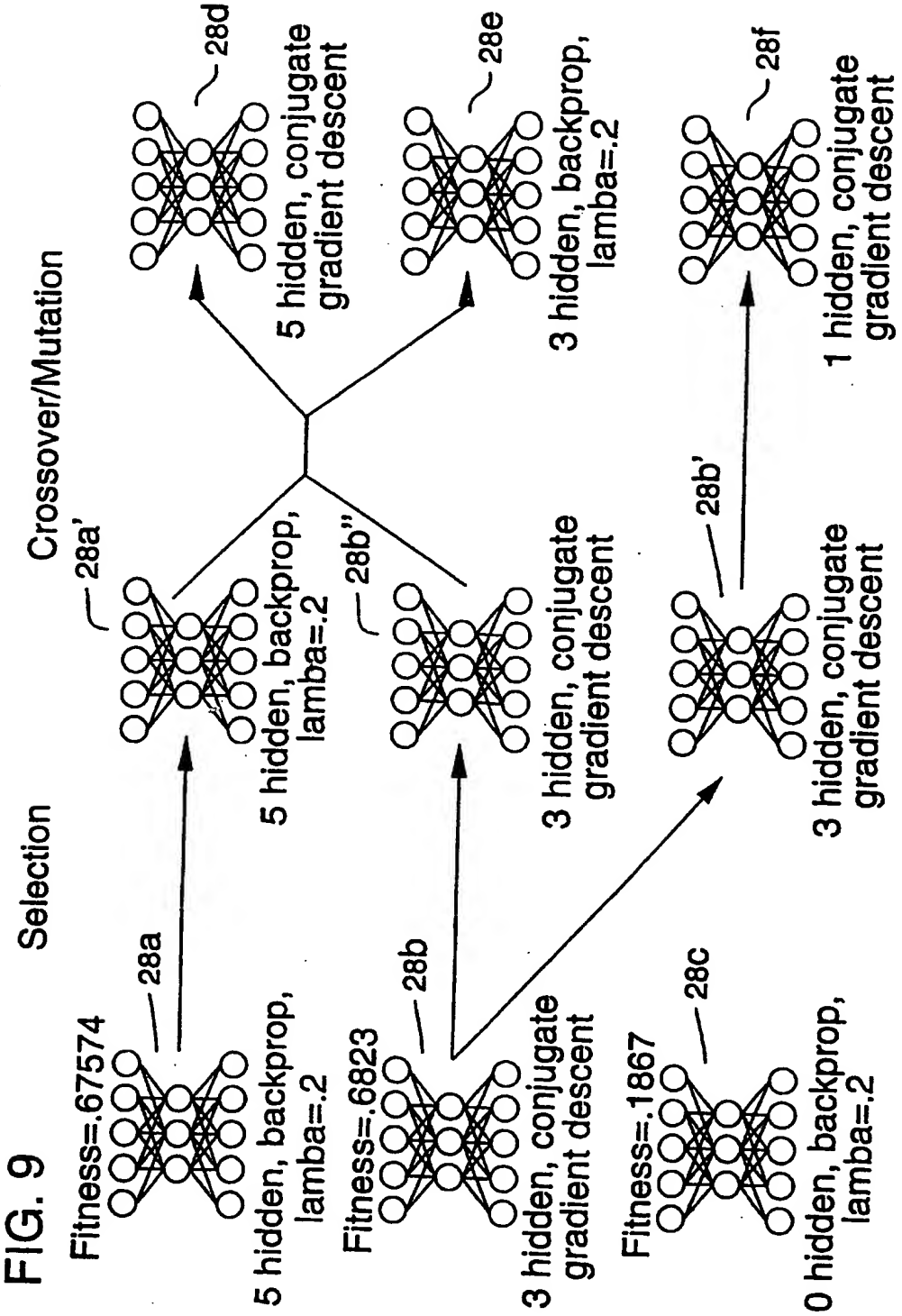

26a
prune=0,
max depth 8
F7,F4*F3,F6>F1
28a
5 hidden,
backprop,lamba=.2
F1,F2/F3,F6 AND F1
28b
0 hidden, backprop,
lamba=.2
F4,F1+F0,F1 OR F1,F2=F3
28c
3 hidden, conjugate,
gradient descent
F1,F3,F4*F5>1,F2>F3
26b
prune=.6,
max depth 4
N-of-M(2,F1,F2,F3),F1>0
26c
prune=0.5,
max depth 3
F1,F3,F4/F2>0,F6=F3
26d
prune=.1,
max depth 8
F1,F2/F3,F6 AND F1
24a
6 variables, quadrarie fit
F1,F2/F3,F6,N-of-M(2,F1,F2,F3),F7,F8
24b
4 variables, least squares
F6 AND F1,F4/F2>0,F6=F3,F7
24c
2 variables, Bayesian
F4*F5,F2/F3
1100|11011
0101|11001
29a
pop size 50, 0.01 mutation rate
F1,F2/F3,F4/F2>0,F6=F3
1100|11011
0101|11001
29b
pop size 20, 0.1 mutation rate
F4/F2>0,F6=F3,F6 AND F1
1100|11011
0101|11001
29c
pop size 100, 0.25 mutation rate
N-of-M(2,F1,F2,F3),F4*F5>1,F2>F3

FIG. 9


Selection
Crossover/Mutation
Fitness=.67574
28a
5 hidden, backprop, lamba=.2
28a'
5 hidden, backprop, lamba=.2
Fitness=.6823
28b
3 hidden, conjugate gradient descent
28b''
3 hidden, conjugate gradient descent
Fitness=.1867
28c
0 hidden, backprop, lamba=.2
28b'
3 hidden, conjugate gradient descent
28d
5 hidden, conjugate gradient descent
28e
3 hidden, backprop, lamba=.2
28f
1 hidden, conjugate gradient descent

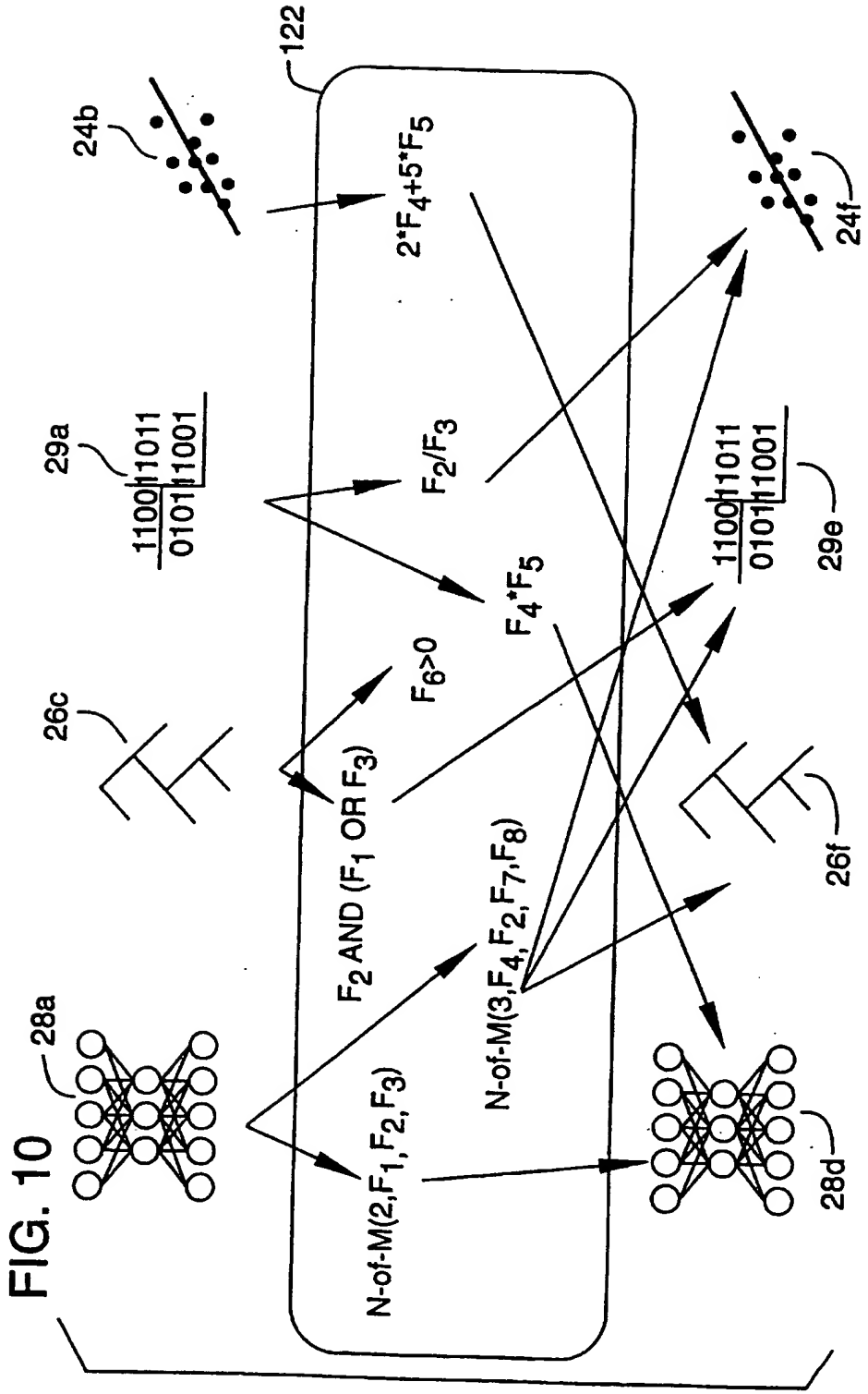

122
24b
24f
29a
1100 1011
0101 1001
29e
26c
26f
28a
28d
$2*F_4+5*F_5$
F_2/F_3
F_4*F_5
$F_6>0$
F_2 AND (F_1 OR F_3)
N-of-M(3,F_4,F_2,F_7,F_8)
N-of-M(2,F_1,F_2,F_3)

FIG. 10

FIG. 11

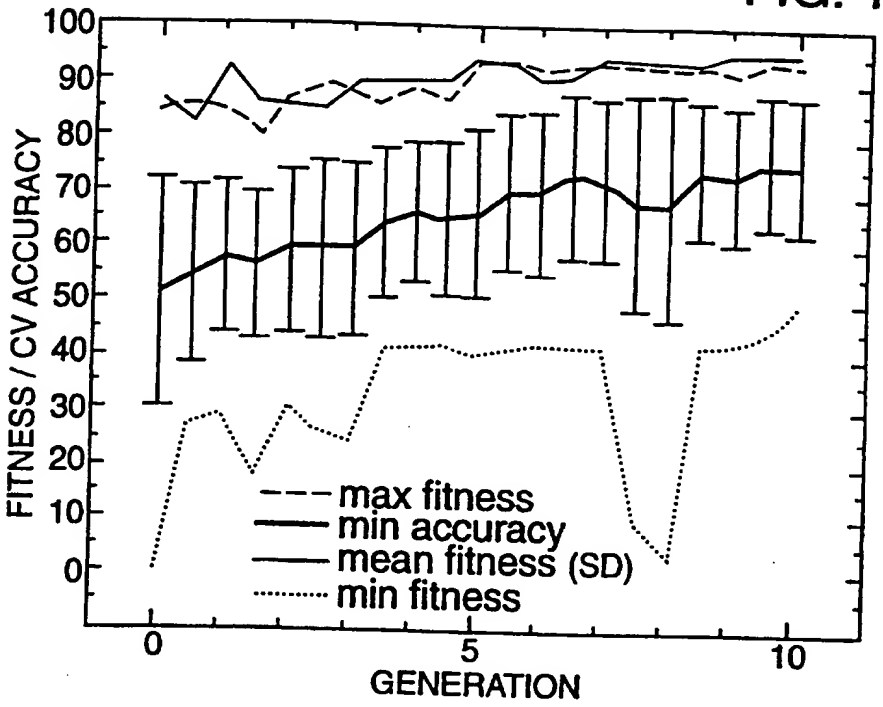

100
90
80
70
60
50
40
30
20
10
0
FITNESS / CV ACCURACY
max fitness
min accuracy
mean fitness (SD)
min fitness
0
5
10
GENERATION

FIG. 12

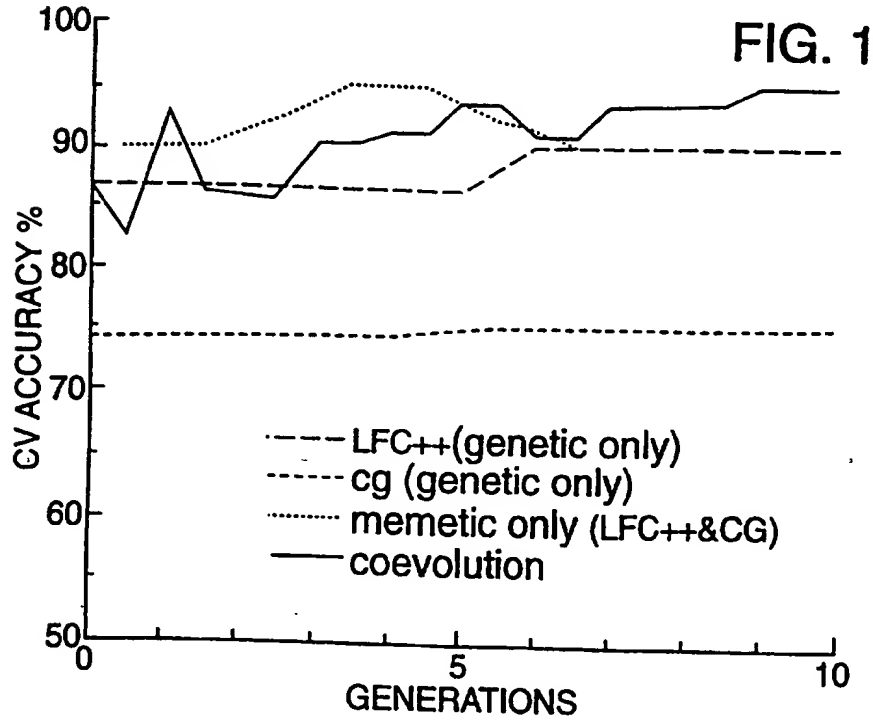
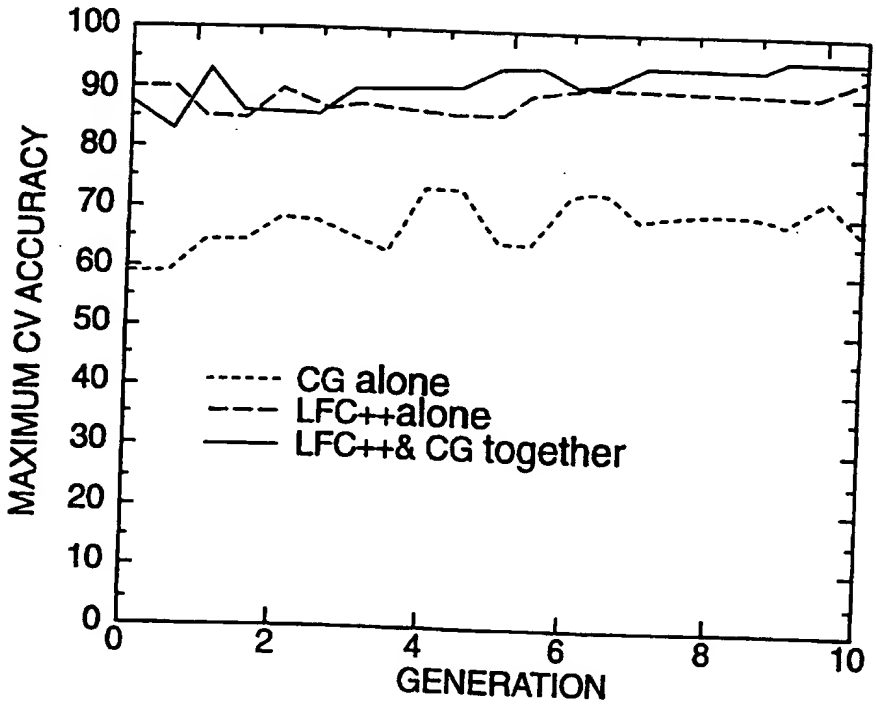

100
90
80
70
60
50
CV ACCURACY %
LFC++(genetic only)
cg (genetic only)
memetic only (LFC++&CG)
coevolution
0
5
10
GENERATIONS

FIG. 13


100
90
80
70
60
50
40
30
20
10
0
MAXIMUM CV ACCURACY
0
2
4
6
8
10
GENERATION
CG alone
LFC++alone
LFC++& CG together

INTERNATIONAL SEARCH REPORT

Intern ial Application No
PCT/US 97/08951

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F15/18

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	BEERUD SHETH ET AL: "EVOLVING AGENTS FOR PERSONALIZED INFORMATION FILTERING" , PROCEEDINGS OF THE CONFERENCE ON ARTIFICIAL INTELLIGENCE FOR APPLICATIONS, ORLANDO, MAR. 1 - 5, 1993, NR. CONF. 9, PAGE(S) 345 - 352 , INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS XP000379626 see page 346, left-hand column, line 53 - page 348, left-hand column, line 52 --- -/--	1-13

[X] Further documents are listed in the continuation of box C.

[ ] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

Date of the actual completion of the international search	Date of mailing of the international search report
10 October 1997	31.10.97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Kingma, Y

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Interna al Application No
PCT/US 97/08951

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	SILVER B ET AL: "ILS: A SYSTEM OF LEARNING DISTRIBUTED HETEROGENEOUS AGENTS FOR NETWORK TRAFFIC MANAGEMENT" PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC), GENEVA, MAY 23 - 26, 1993, vol. 3 OF 3, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, pages 1869-1874, XP000448444 see abstract ---	5
A	FAWCETT T E ET AL: "A hybrid method for feature generation" , MACHINE LEARNING. PROCEEDINGS OF THE EIGHTH INTERNATIONAL WORKSHOP ON MACHINE LEARNING (ML91), EVANSTON, IL, USA, JUNE 1991 , 1991, SAN MATEO, CA, USA, MORGAN KAUFMANN, USA, PAGE(S) 137 - 141 XP002043275 see page 138, right-hand column, line 19 - line 44 -----	8

Form PCT/ISA/210 (continuation of second sheet) (July 1992)